

Materialized Views

Scalable Data Engineering

Motivation

Multidimensional Modeling

- Conceptual model data cube
- Multidimensional Operators „Roll Up“

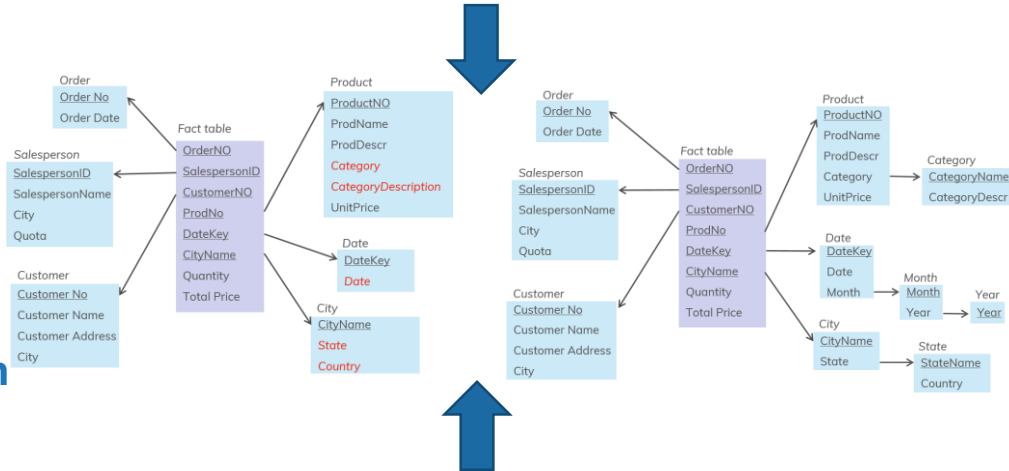
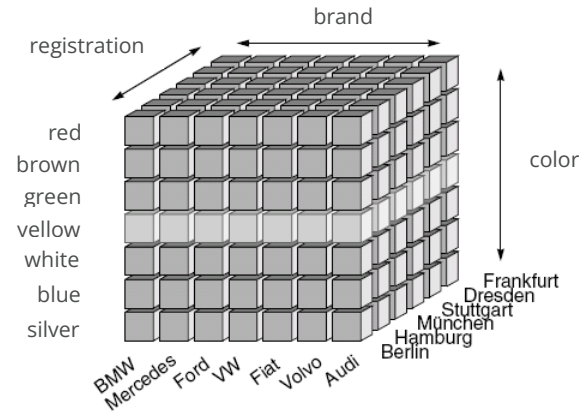
Relational Mapping

- Star Schema
- Snowflake Schema

Query Processing

- Complex Queries and Calculations
 - Maybe SQL/OLAP extensions

Optimization of read or calculation intensive work loads.



```
SELECT Year, SUM(Total Price)
FROM Fact Table
GROUP BY Year
```

Agenda

Motivation und Terminology

Selection of Materialized Views

- Automatic Selection

Usage of Mat. Views

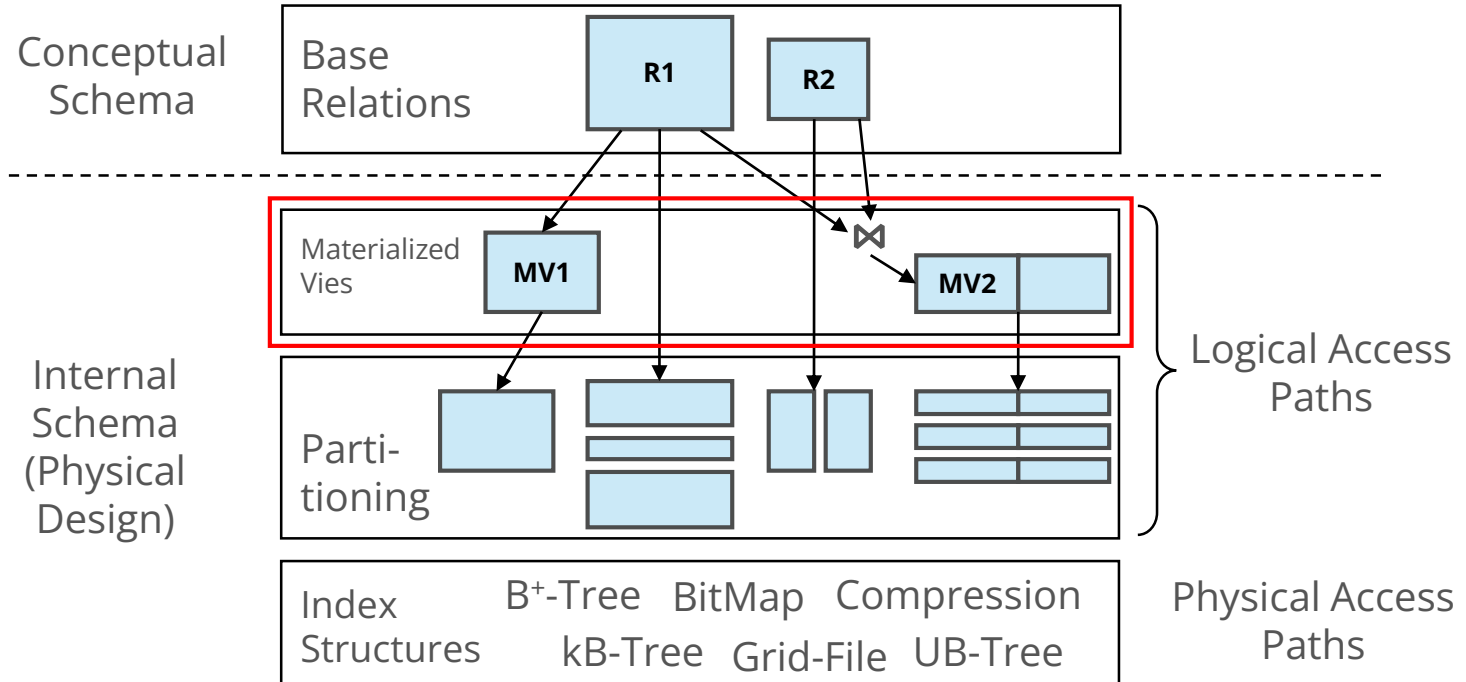
- Compensation Queries

Mat. View Maintenance

- Triggering Maintenance
- Maintenance Strategies

Motivation Optimization

Overview Optimization

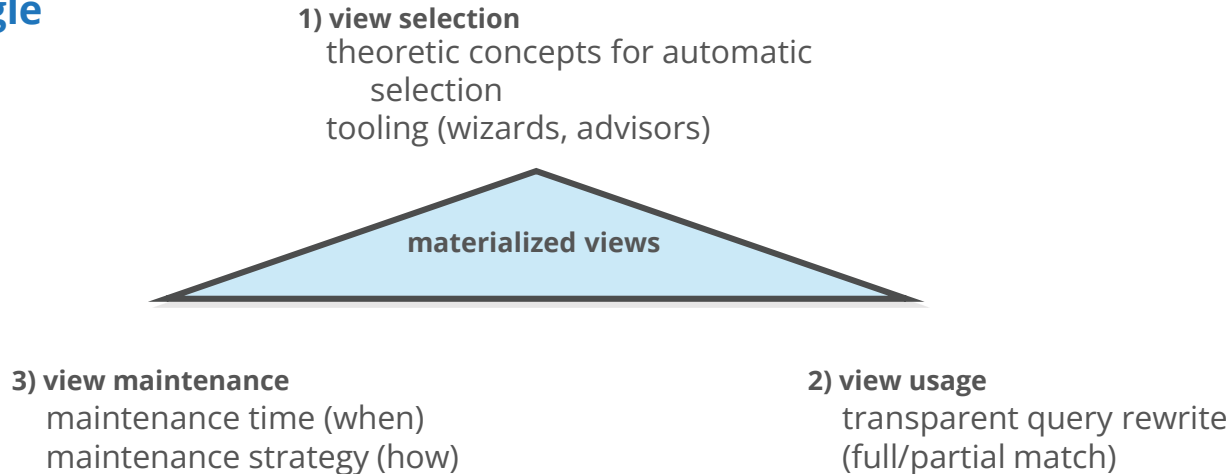


Basic Idea of Materialized Views

Basics

- Identification of frequently reoccurring queries
- Precalculate and store it
- Repeated use of the precalculated query

The Magical Triangle



Types of Materialized Views

- Materialized join view
- Materialized aggregate view
- Materialized aggregate-join view

View Selection / View Creation

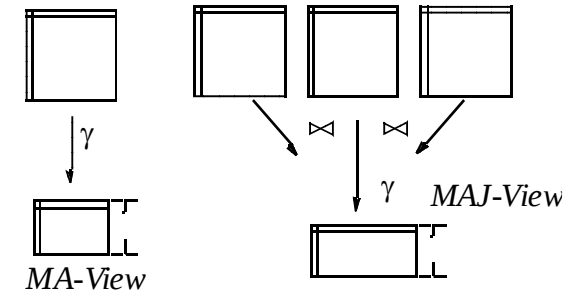
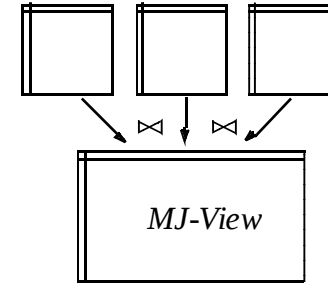
- High number of grouping attribute combinations in a multidimensional DWH
- Which combination is optimal regarding a workload and a limited storage budget

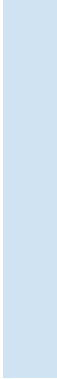
View Usage / View Matching

- Transparent usage analogously to index structures

View Maintenance / View Update

- Synchronization of mat. view when base data changes





Selection of Materialized Views

Derivability of queries

- Query Q may be replaced by Q'
- Q' uses the contents of mat. view V
- Q' delivers a semantically equivalent results to Q
- Q' is called compensation query

How to construct a mat. view

- What should a partial/complete replacement of a query by a mat. view look like
- More effective when a mat. view is used for several replacements

Identification of derivability is NP-hard

- Existence statement

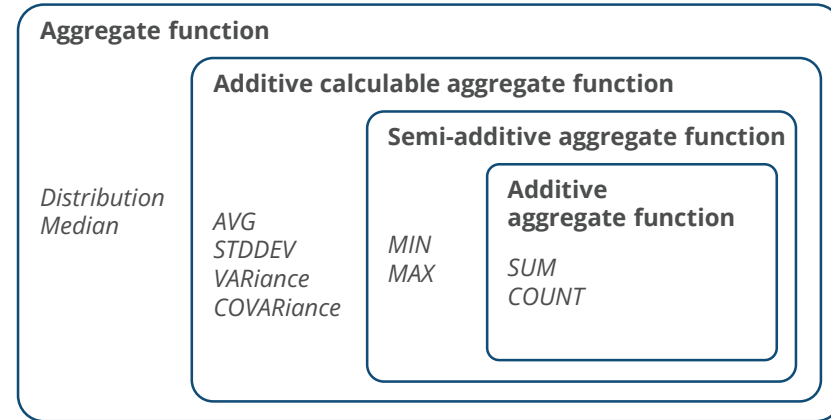
Range limits

- Predicate P_M und P_Q : $P_Q \subseteq P_M$
 - General undecidability of predicate logic
 - Reduction to elemental predicates after standardization in disjunctive normal form

Classification of Aggregate Functions

Compatibility of aggregate functions

- Additive aggregate functions
 - $F()$ is a cumulative group regarding the facts and measures of the data cube $F(x_1 \cup x_2) = F(\{F(x_1), F(x_2)\})$ and $F^{-1}()$ exists
 - Pay attention to summation type
 - Example:
 - $SUM(x_1 \cup x_2) = SUM(\{SUM(x_1), SUM(x_2)\})$ and
 - $SUM(x_1) = SUM(x_1 \cup x_2) - SUM(x_2)$
- Semi-additive aggregate functions
 - The inverse is missing
 - No limitations regarding derivability
 - No incremental maintenance regarding delete operations



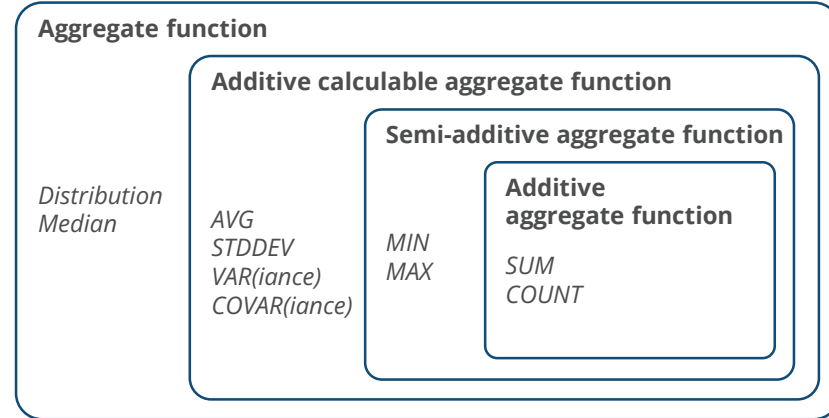
Classification of Aggregate Functions

Compatibility of aggregate functions

- Additive calculable aggregate functions
 - Algebraic expression $G()$ over semi-additive or additive aggregate function $F_1(), \dots, F_n()$ exist, s.t., $F(x) = G(F_1(x), \dots, F_n(x))$ holds
 - Example:

$$AVG(x) = \frac{SUM(x)}{COUNT(x)}$$

$$COVAR(x, y) = \frac{SUM(x \cdot y - \left(\frac{SUM(x) \cdot SUM(y)}{COUNT(*)}\right))}{COUNT(*) - 1}$$



Direct derivability of grouping attribute combinations

- Two grouping attribute combinations G_1 and G_2
- G_2 is directly derivable from G_1 when:
 - Grouping attribute combination G_2 has one attribute less than grouping attribute combination G_1 , i.e., $G_2 \subset G_1$ and $|G_2| = |G_1| - 1$ (or)
 - One of the attributes A_i' in the grouping attribute combination G_2 is functionally defined by A_i in G_1 , i.e., $A_i \rightarrow A_i'$

Examples

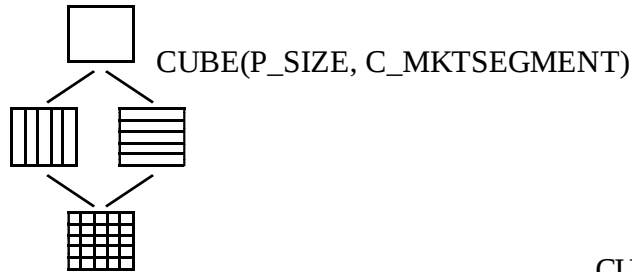
- G_2 : (P_SIZE, C_MKTSEGMENT) from G_1 : (P_SIZE, C_MKTSEGMENT, L_SHIPINSTRUCT)
- G_2 : (P_SIZE, R_NAME) from G_1 : (P_SIZE, N_NATION)

Derivability of grouping attribute combinations

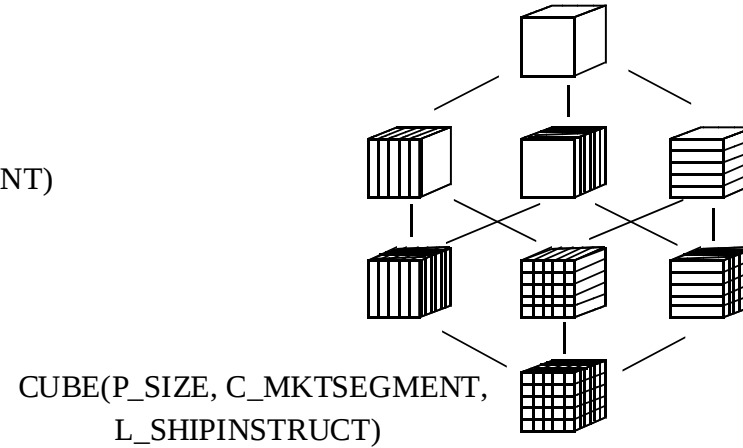
- Grouping attribute combinations G_2 is derivable from G_1 inside of an aggregation grid, when there is a path of direct derivabilities

Grouping attribute combinations for 2^n attributes

- See aggregation grids for CUBE operator



Two attributes $2^2=4$ combinations



Three attributes $2^3=8$ combinations

Mat. view selection

- Requirements
 - Aggregation grid (Type, Orderstatus, Marktsegment)
- Classes of aggregate functions
- Cost model for benefit and update

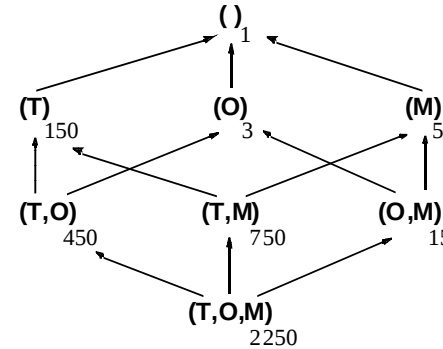
Selection Algorithm

- Problem is NP-hard (for the general case)
- Greedy-algorithm with and without update cost

[Wolfgang Lehner: Datenbanktechnologie für Data-Warehouse-Systeme: Konzepte und Methoden, dpunkt-Verlag, 2003.]

- Genetic selection algorithm for distributed case

[Leonardo Weiss Ferreira Chaves, Erik Buchmann, Fabian Hueske, Klemens Böhm: Towards materialized view selection for distributed databases. EDBT 2009]



Selection Algorithm

Preconditions

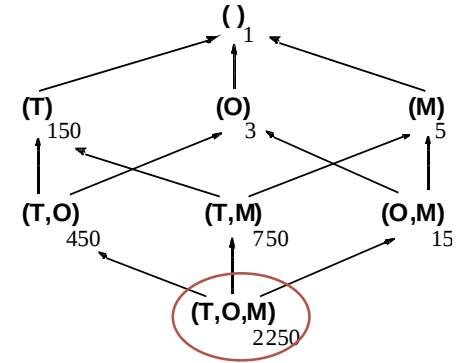
- Aggregation grid
- Storage limit
- Cost model (tuple count for sake of simplicity)
- Materialization of base granularity (T,O,M)

Benefit calculation

- $B(g) = d(g) \cdot (c(base) - c(g))$
 - $c()$ – cost of grouping
 - $d()$ – no. of derivable groupings, e.g., $d((T,O)) = \{(T,O), (T), (O), ()\}$

Benefit per storage

- $BPS(g) = B(g)/c(g)$



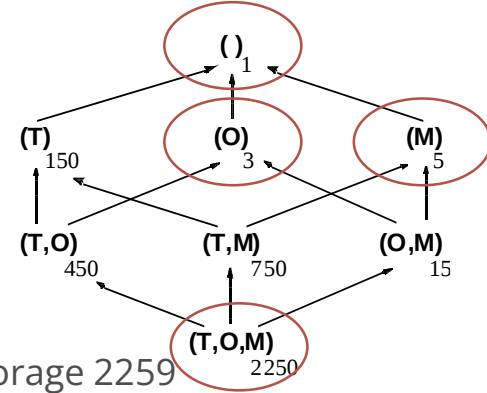
Selection Algorithm

Iterative approach

- Check BPS of each grouping in every Iteration
- Storage limit: 2300

$$B(g) = d(g) \cdot (c(base) - c(g))$$

$$BPS(g) = B(g)/c(g)$$



1st Iteration

Storage 2251

g	B	BPS
(T,O)	4*(2250-450)=7200	16
(T,M)	4*(2250-750)=6000	8
(O,M)	4*(2250-15)=8940	596
(T)	2*(2250-150)=4200	28
(O)	2*(2250-3)=4494	1498
(M)	2*(2250-2)=4490	898
()	1*(2250-1)=2249	2249

2nd Iteration

Storage 2254

g	B	BPS
(T,O)	3*1800=5400	12
(T,M)	3*1500=4500	6
(O,M)	3*2235=6705	447
(T)	1*2100=2100	14
(O)	1*2247=2247	749
(M)	1*2245=2245	449

3rd Iteration

Storage 2259

g	B	BPS
(T,O)	2*1800=3600	8
(T,M)	3*1500=4500	6
(O,M)	2*2235=4470	298
(T)	1*2100=2100	14
(M)	1*2245=2245	449

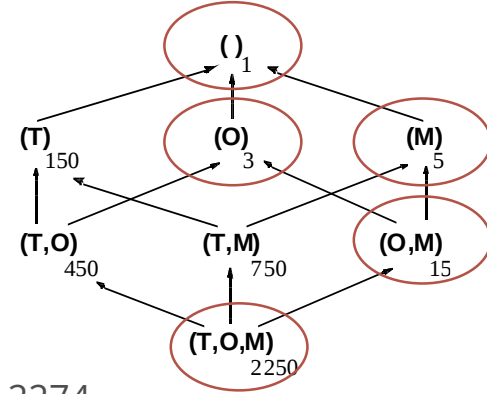
Selection Algorithm

Iterative approach

- Check BPS of each grouping in every Iteration
- Storage limit: 2300

$$B(g) = d(g) \cdot (c(base) - c(g))$$

$$BPS(g) = B(g)/c(g)$$



2nd Iteration Storage 2254

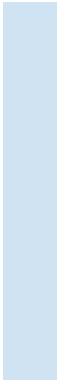
g	B	BPS
(T,O)	3*1800=5400	12
(T,M)	3*1500=4500	6
(O,M)	3*2235=6705	447
(T)	1*2100=2100	14
(O)	1*2247=2247	749
(M)	1*2245=2245	449

3rd Iteration Storage 2259

g	B	BPS
(T,O)	2*1800=3600	8
(T,M)	3*1500=4500	6
(O,M)	2*2235=4470	298
(T)	1*2100=2100	14
(M)	1*2245=2245	449

4th Iteration Storage 2274

g	B	BPS
(T,O)	2*1800=3600	8
(T,M)	2*1500=3000	4
(O,M)	1*2235=2235	149
(T)	1*2100=2100	14



Usage of Mat. Views

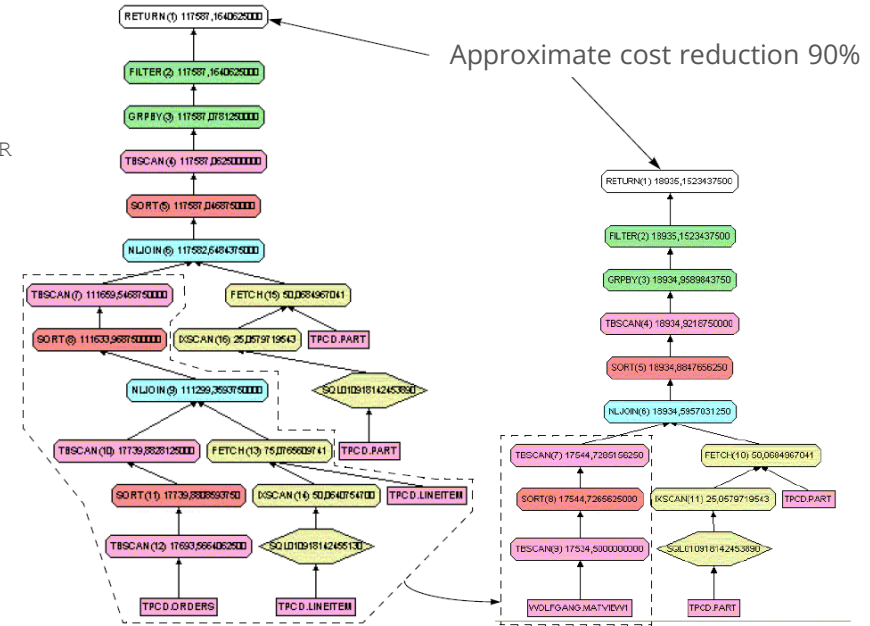
Usage Example

Mat. view definition (DB2)

```
CREATE SUMMARY TABLE MATVIEW1 AS (
  SELECT L_PARTKEY, O_ORDERPRIORITY, O_ORDERSTATUS,
         SUM(L_QUANTITY) AS SUM_QUAN,
         SUM(L_QUANTITY * L_EXTENDEDPRICE) AS FULL_TURNOVER
  FROM TPCD.LINEITEM, TPCD.ORDERS
  WHERE L_ORDERKEY = O_ORDERKEY
  GROUP BY L_PARTKEY, O_ORDERPRIORITY, O_ORDERSTATUS
) DATA INITIALLY DEFERRED REFRESH IMMEDIATE;
```

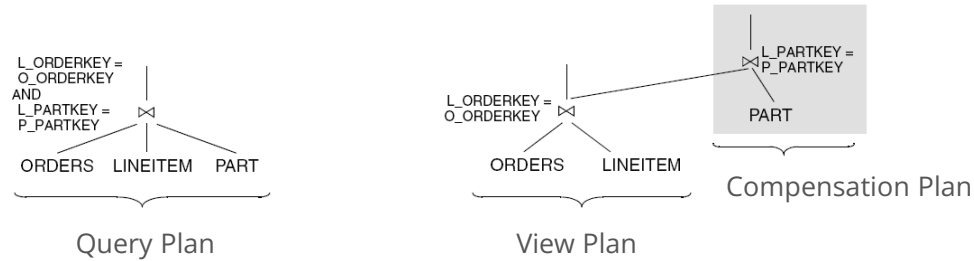
Example Query

```
SELECT P_BRAND, O_ORDERSTATUS,
       SUM(L_QUANTITY * L_EXTENDEDPRICE)
  AS TURNOVER
  FROM TPCD.LINEITEM, TPCD.ORDERS, TPCD.PART
  WHERE L_ORDERKEY = O_ORDERKEY
  AND L_PARTKEY = P_PARTKEY
  AND O_ORDERPRIORITY = '2-HIGH'
  AND P_CONTAINER = 'LG_BAG'
  GROUP BY P_BRAND, O_ORDERSTATUS
  HAVING AVG(L_QUANTITY) > 250;
```

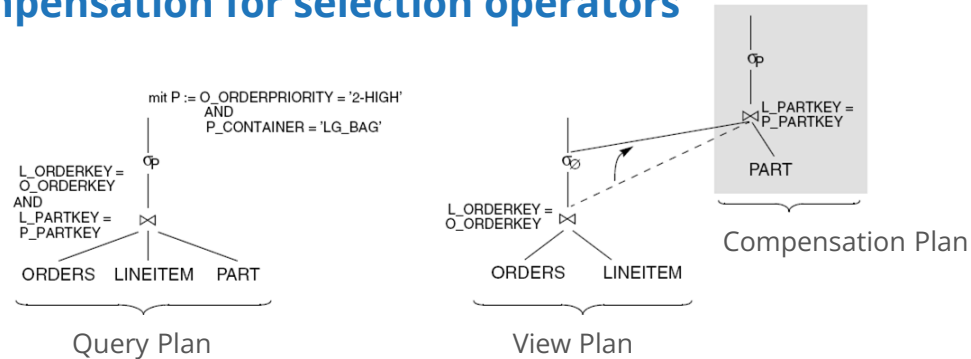


Usage Example

Derivability and compensation for join operators

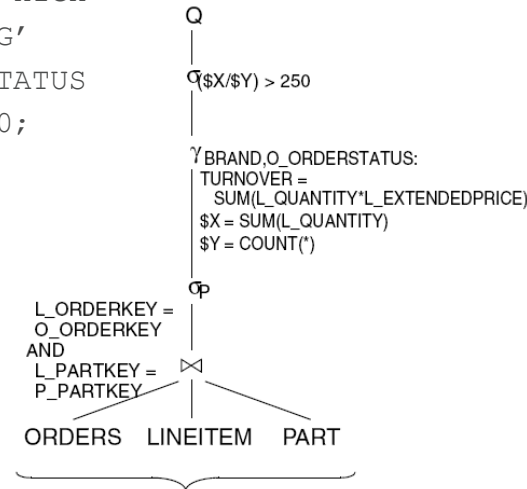


Derivability and compensation for selection operators

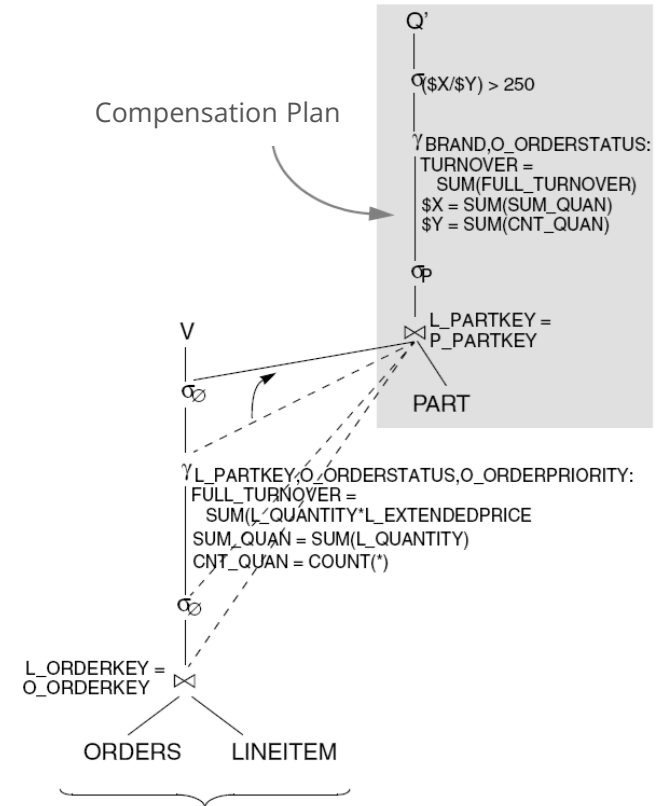


Usage Example

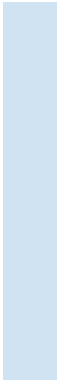
```
SELECT P_BRAND, O_ORDERSTATUS,
       SUM(L_QUANTITY * L_EXTENDEDPRICE)
       AS TURNOVER
FROM LINEITEM, ORDERS, PART
WHERE L_ORDERKEY = O_ORDERKEY
      AND L_PARTKEY = P_PARTKEY
      AND O_ORDERPRIORITY = '2-HIGH'
      AND P_CONTAINER = 'LG_BAG'
GROUP BY P_BRAND, O_ORDERSTATUS
HAVING AVG(L_QUANTITY) > 250;
```



Query Plan



View Plan



Mat. View Maintenance

Problems of explicit redundancy

- Mat. views require maintenance
- Transparent for DB users

Maintenance terminology

- Autonomous maintainability or „self maintenance“
 - A mat. view is autonomously maintainable if the following are sufficient to update the view
 - Changes caused by INSERT, UPDATE, and DELETE operations on base relations
 - View definition
 - Current content of the view
 - Partial autonomy using schema information, helper views or counting variables

Time of Maintenance (when)

- Eager vs. Deferred (vs. Lazy)

Maintenance strategy (how)

- Complete vs. Incremental

Immediate maintenance („immediate refresh“)

- Changes are forwarded to the mat. view during the pending operation
- Following operations of this transaction see a consistent state
- Complete rollback if transaction fails
- Example DB2: REFRESH IMMEDIATE clause when defining the mat. view

Transactional maintenance („on commit refresh“)

- Trigger maintenance on commit of the changing transaction
- Possible access on inconsistent information during this transaction
- Example Oracle9i: ON COMMIT parameter

Time of Maintenance - Deferred

Deferred maintenance („deferred refresh“)

- Maintenance only when the mat. view is used
- No consistency guaranties between base relation and mat. view
- Examples DB2: REFRESH DEFERRED, Oracle9i: ON DEMAND
- Reading queries pay the update cost (but they actually should benefit)

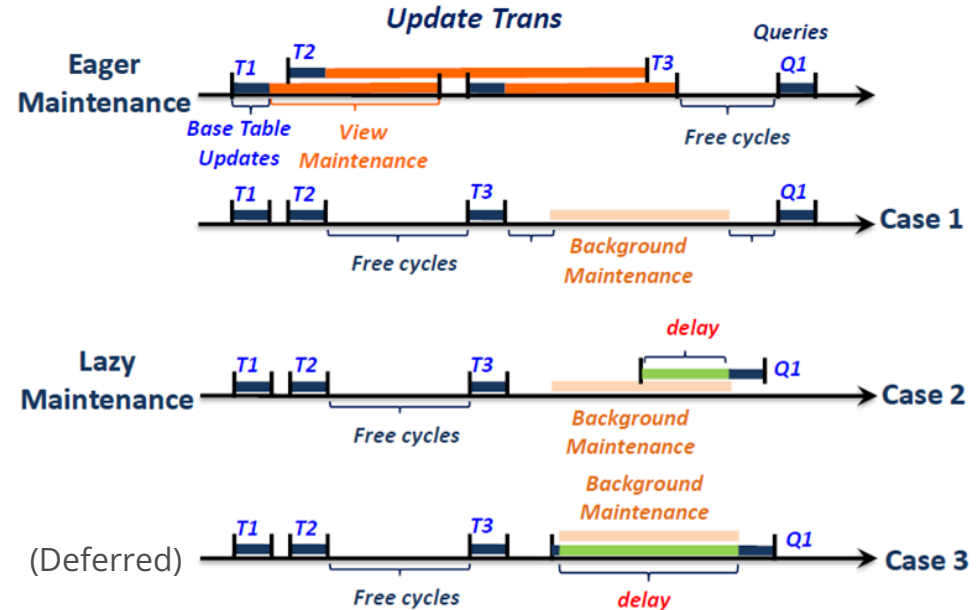
Time of Maintenance - Lazy

Motivation

- Queries should profit
- Updating transaction pay for the maintenance (of possibly unused views)

Approach

- Deferred maintenance except:
 - There are free cycles
- Maintenance is invisible for updaters and queries
- Efficient Maintenance



Maintenance Strategy

Orthogonal to time of maintenance

Incremental maintenance

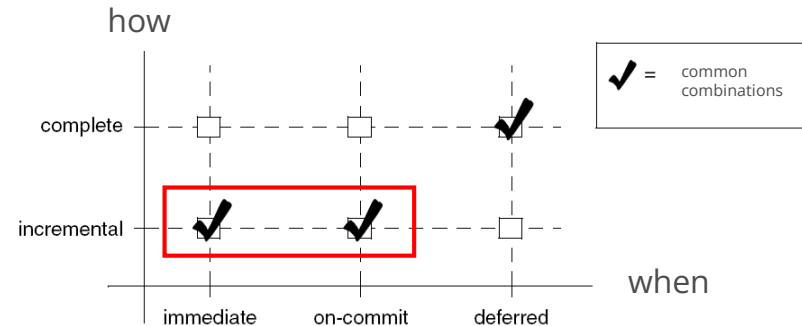
- Require autonomous maintainability („self maintainability“)
- Changes on base relations are sufficient for mat. view maintenance

Complete maintenance/rebuild

- Complete recalculation of the mat. view

Possible combinations of

- How and when



Overview Incremental Maintenance

Incremental update

- Propagation phase: produce/collect changes
- Apply phase: bring changes to the mat. view

View definition:
SELECT A, SUM(B)
FROM Sales
GROUP BY CUBE(A)

A	SUM
NULL	107
X	30
Y	77



Local Changes
 ΔR



Local
View Delta
 ΔV_L



Global
View Delta
 ΔV_G



Super Delta
 ΔV_S



Update View
 ΔV_A

A	B
+ X	3
+ Z	9

A	SUM
+ NULL	3
+ X	3
+ NULL	9
+ Z	9

A	SUM
+ NULL	12
+ X	3
+ Z	9

A	SUM	SUM2
NULL	12	107
X	3	30
Z	9	NULL

A	SUM
Update NULL	119
Update X	33
Insert Z	9

Propagate phase

Apply phase

Example

View definition

```
SELECT P_BRAND, MONTHNAME(O_ORDERDATE) AS O_ORDERMONTH,
       SUM(L_QUANTITY) AS SUM_QUANTITY, COUNT(*) AS CNT_QUANTITY
FROM LINEITEM, ORDERS, PART
WHERE L_ORDERKEY = O_ORDERKEY
      AND L_PARTKEY = P_PARTKEY
      AND P_CONTAINER = 'LG BAG'
      AND YEAR(O_ORDERDATE) = 2002
GROUP BY CUBE(P_BRAND, MONTHNAME(O_ORDERDATE));
```

Local changes

- -1 / +1 summary of changes in the base data (out/in)

L_ORDERKEY	L_PARTKEY	L_SUPPKEY	...	L_QUANTITY	...	O_ORDERDATE	...	P_BRAND	P_TYPE	...	L_TAG
-----	-----	-----	...	-----	...	-----	...	-----	-----	...	-----
512	43	323	...	149	...	2002-02-13	...	Brand#23	LG BAG	...	-1
512	43	323	...	156	...	2002-02-13	...	Brand#23	LG BAG	...	+1
734	21	311	...	48	...	2002-03-23	...	Brand#02	LG BAG	...	+1
392	86	483	...	211	...	2002-03-23	...	Brand#46	MED BAG	...	+1
231	23	933	...	76	...	2002-03-23	...	Brand#23	LG BAG	...	+1

Local View Delta

- Application of view definition on local changes
- Result: summation data from local changes
 - Distinguished into inserts and deletes L_TAG
- Saved in an explicit log table

O_ORDERMONTH	P_BRAND	SUM_QUANTITY	CNT_QUANTITY	L_TAG
February	Brand#23	149	1	-1
February	NULL	149	1	-1
NULL	Brand#23	149	1	-1
NULL	NULL	149	1	-1
February	Brand#23	156	1	+1
March	Brand#23	76	1	+1
March	Brand#02	48	1	+1
February	NULL	156	1	+1
March	NULL	124	2	+1
NULL	Brand#23	232	2	+1
NULL	Brand#02	48	1	+1
NULL	NULL	280	3	+1

} ==> \$log

Global View Delta

- Summary of accumulated changes
- `L_TAG` required to ensure correct results

```
SELECT P_BRAND, O_ORDERMONTH,  
       SUM(SUM_QUANTITY*L_TAG) AS SUM_QUANTITY,  
       SUM(CNT_QUANTITY*L_TAG) AS CNT_QUANTITY  
FROM $log           // log table  
GROUP BY P_BRAND, O_ORDERMONTH
```

O_ORDERMONTH	P_BRAND	SUM_QUANTITY	CNT_QUANTITY	} ==> \$viewdelta
February	Brand#23	7	0	
March	Brand#23	76	1	
March	Brand#02	48	1	
February	NULL	7	0	
March	NULL	124	2	
NULL	Brand#23	83	1	
NULL	Brand#02	48	1	
NULL	NULL	131	2	

Super Delta

- Join global view delta with current state of the mat. view
- Left-out-join: delta tuples which create a new group must not be lost

```
SELECT P_BRAND, O_ORDERMONTH,  
       DELTA.SUM_QUANTITY, DELTA.CNT_QUANTITY  
FROM $viewdelta DELTA LEFT OUTER JOIN  
     $matview MV ON JOINPRED(P_BRAND, O_ORDERMONTH)
```

- Join predicate:

```
( (DELTA.P_BRAND = MV.P_BRAND) OR  
  (DELTA.P_BRAND IS NULL AND MV.P_BRAND IS NULL) ) AND  
( (DELTA.O_ORDERMONTH = MV.O_ORDERMONTH) OR  
  (DELTA.O_ORDERMONTH IS NULL AND MV.O_ORDERMONTH IS NULL) ) ...
```

O_ORDERMONTH	P_BRAND	DELTA.SUM_QUANTITY	DELTA.CNT_QUANTITY	MV.SUM_QUANTITY	MV.CNT_QUANTITY
February	Brand#23	7	0	431	5
March	Brand#23	76	1	NULL	NULL
March	Brand#02	48	1	NULL	NULL
February	NULL	7	0	10232	122
March	NULL	124	2	NULL	NULL
NULL	Brand#23	83	1	4233	54
NULL	Brand#02	48	1	4122	48
NULL	NULL	131	2	154344	855

Update view

- Columns of delta and mat. view are numerically connected
- Example

O_ORDERMONTH	P_BRAND	DELTA.SUM_QUANTITY_NEW	DELTA.CNT_QUANTITY_NEW	MV.QUANTITY_CNT
February	Brand#23	438	0	5
March	Brand#23	76	1	NULL
March	Brand#02	48	1	NULL
February	NULL	10239	122	122
March	NULL	124	2	NULL
NULL	Brand#23	4316	55	54
NULL	Brand#02	4170	49	48
NULL	NULL	154475	857	855

Apply changes

- Depending on old value of count variable in mat. view
 - NULL value: INSERT operation
 - NOT NULL: UPDATE operation, maybe DELETE, if new mat. view value is 0

Requirements for View Maintenance

Limits on aggregate function classes

- Limited to additive aggregate functions
- Semi-additive functions are possible if incremental maintenance is not necessary (for delete)

Existence of grouping clauses

- All grouping attributes have to be used in one complex grouping clause, s.t., natural NULL values and generated NULL values can be distinguished

Existence of count variables

- COUNT(*)/COUNT(X) attributes in view definition to manage the maintenance

Distinct grouping combinations

- Arbitrary complex grouping clauses, as long as each combination is unique
- Example: GROUP BY ROLLUP(B,A),A = GROUPING SETS ((B,A,A)=(**B,A**), (**B,A**), (A)) -> invalid

More limitations regarding operations and the usage of several relations

Summary

Motivation und Terminology

Selection of Materialized Views

- Automatic Selection

Usage of Mat. Views

- Compensation Queries

Mat. View Maintenance

- Triggering Maintenance
- Maintenance Strategies

Exercise Preparation

Recap the exercise on Multidimensional Queries

- Install PostgreSQL
- Think about what makes queries expensive

Visit the PostgreSQL documentation

- Create Mat. View syntax
 - <https://www.postgresql.org/docs/14/sql-creatematerializedview.html>
- Mat. View documentation
 - <https://www.postgresql.org/docs/14/rules-materializedviews.html>